МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №3
по курсу «Программирование графических процессоров»

Классификация и кластеризация изображений на GPU.

Выполнил: И.Т. Батяновский
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы: Научиться использовать GPU для классификации и кластеризации изображений. Использование константной памяти.

Вариант 5. Метод Робертса.

Программное и аппаратное обеспечение

Device: GeForce GTX 970
Размер глобальной памяти: 4294967296
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 65536
Максимум потоков на блок: 1024
Количество мультипроцессоров : 13
OS: Windows 10
Редактор: CLion

Метод решения

Инициализируются количество и центры кластеров. Так как центры кластеров будут часто вызваться имеет смысл занести их в константную память, что положительно скажется на производительности. Далее для каждого пикселя рассчитывается расстояния до кластеров и выбирается кластер с минимальным расстоянием. Счетчик на количество элементов в данном кластере увеличивается. После всех подсчетов центры кластеров смещаются.

$$\mu_i^{(n+1)} = \frac{1}{\left|S_i\right|} \sum_{f \in S_i} f$$

Если центры не изменились, то процесс завершается.

Описание программы

Считываются данные и создается массив:

```
fin.read((char *) &width, 4);
fin.read((char *) &height, 4);

ImageType *host_data = new ImageType [width * height];
fin.read((char *) host_data, sizeof(ImageType) * height * width);
```

Инициализируются количество и центры кластеров.

```
int clastersNum;
cin >> clastersNum;

ClastersPos * host_positions = new ClastersPos [clastersNum];
for (int i = 0, x, y; i < clastersNum; i++)
{
   cin >> x >> y;

   host_positions[i].x = host_data[y * width + x].x;
   host_positions[i].y = host_data[y * width + x].y;
   host_positions[i].z = host_data[y * width + x].z;
}
```

Количество кластеров и их центры храня в константной памяти.

```
__constant__ int QUANTITY;
__constant__ ClastersPos POSITIONS[32];

cudaMemcpyToSymbol(QUANTITY, &clastersNum, sizeof(int));
cudaMemcpyToSymbol(POSITIONS, host_positions, clastersNum * sizeof(ClastersPos));
```

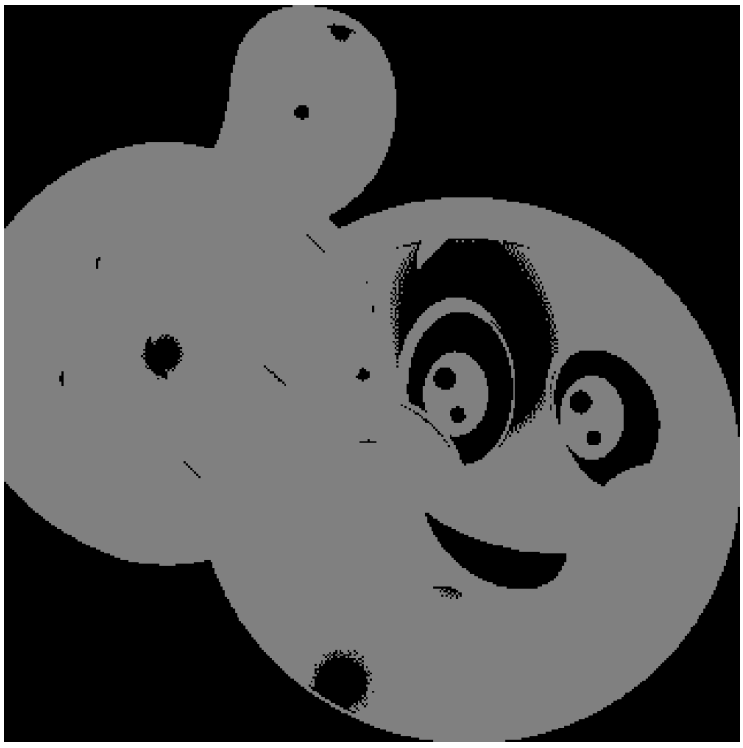
Далее запускается kernel, которая просто находит кластер с ближайшим центром.

```
__global__ void kernel(ImageType *data, int n, int m)
{
   // Calculate normalized texture coordinates
```

```
    int x = blockIdx.x * blockDim.x + threadIdx.x;
    int y = blockIdx.y * blockDim.y + threadIdx.y;
    int xOffset = blockDim.x * gridDim.x;
    int yOffset = blockDim.y * gridDim.y;

    for (int i = y; i < m; i += yOffset)
    {
        for (int j = x; j < n; j += xOffset)
        {
            ImageType threadValue = data[i * n + j];
            DistanceType minDist = 1.7976931348623158e+308;
            DistanceType tempDist;

            int tempClaster;

            for (int claster = 0; claster < QUANTITY; claster++)
            {
                tempDist = dist(threadValue, POSITIONS[claster]);

                if (tempDist < minDist)
                {
                    minDist = tempDist;
                    tempClaster = claster;
                }
            }

            data[i * n + j].w = tempClaster;
        }
    }
}
```

Далее на процессоре происходит подсчет элементов кластеров и смещение центров. Если центр не изменились программа завершается и выводит матрицу в файл.

Результаты



В изображении присутствуют два кластера с центрами в верхнем левом углу в центре. На лицо некоторые неточности, которые устраняются с добавлением новых классов и новых точек для них.

Таблица замеров времени выполнения

Threads, size	ms
16 * 16	0.012
32 * 32	0.015
64 * 64	0.003
128 * 128	0.001
256 * 256	0.001

Время работы с разным количеством запущенных потоков.

Выводы

Во время выполнения я пользовался константной памятью и убедился, что она является достаточно быстрой из доступных GPU. Если необходимо использовать массив в константной памяти, то его размер необходимо указать заранее, так как динамическое выделение не поддерживается. Возникли трудности вначале трудности со смещением так как хотел использовать возможности cuda, но в итоге сам себя запутал. Более простая реализация на процессоре решила проблему.